Performance Evaluation of a Bluetooth Channel Estimation Algorithm

N. Golmie
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Abstract— **Since Bluetooth devices have to share the already crowded unlicensed ISM band with WLAN spread spectrum devices that may be operating in close proximity, a mechanism that detects interference and avoids transmission in the so-called "bad" channels is key in order to prevent mutual interference and the resulting performance degradation. In this paper, we describe a dynamic channel estimation algorithm and evaluate its performance in conjunction with a Bluetooth scheduling algorithm that avoids transmission in frequencies used by other devices. Simulation results for several scenarios of interest including different traffic types are presented and analyzed.**

I. Introduction

Bluetooth is a short range (0 m - 10 m) wireless personal area network (WPAN) technology aimed at replacing non-interoperable proprietary cables that connect phones, laptops, PDAs and other portable devices together.

Rather than competing with Wireless Local Area Networks (WLANs) for spectrum and applications, WPANs are intented to augment many of the usage scenarios and operate in conjunction with WLANs. However, since both WPANs and WLANs employ the 2.4 GHz ISM band, an issue of growing concern is coexistence of these technologies in the same environment. Several techniques and algorithms aimed at reducing the impact of interference have been considered. These techniques range from collaborative schemes intended for Bluetooth and IEEE 802.11 protocols to be implemented in the same device to fully independent solutions that rely on interference detection and estimation. Mechanisms for collaborative schemes have been proposed to the IEEE 802.15 Coexistence Task Group and are based on a MAC time domain solution that alternates the transmission of Bluetooth and WLAN packets (assuming both protocols are implemented in the same device and use a common transmitter) [1]. A priority of access is given to Bluetooth for transmitting voice packets, while WLAN is given priority for transmitting data. On the other hand, non-collaborative mechanisms range from adaptive frequency hopping [2] to packet scheduling and traffic control [3]. If frequency hopping devices know which frequencies are occupied by other users of the band, they can modify their frequency hopping pattern in order to avoid transmission in so-called "bad" frequencies. They can even choose not to transmit on certain frequencies without modifying their adaptive frequency hopping sequence. The first technique is known as adaptive frequency hopping, while the second technique is known as MAC scheduling. The main advantage of scheduling over adaptive hopping is that scheduling does not require any changes to the Bluetooth specifications.

Both adaptive frequency hopping and scheduling, rely on channel estimation techniques in order to detect the presence of other devices in the band. These techniques are based on measurements performed on each frequency such as Bit Error Rate (BER), Frame Error Rate (FER), the signal strength, or the signal to interference ratio (SIR) (also known as the Received Signal Strength Indicator (RSSI)). One or several of these measurements constitute the basis criteria for marking a frequency as "bad". In this paper, we focus on a dynamic channel estimation procedure and evaluate its responsiveness to interference environments where connections are set up and torn down over time. In addition, we look at different traffic types including MP3, voice, ftp and http.

This paper is organized as follows. In sections II, we discuss the details of the dynamic channel estimation procedure and overview the Bluetooth scheduling mechanism used. In section III, we give simulation results and concluding remarks are offered in section IV.

II. Interference Estimation and Bluetooth Scheduling

In this section, we describe a channel estimation procedure to be implemented in Bluetooth in order to detect the presence of other wireless devices in the band. This technique is most effective for detecting other slower hopping (or even non-hopping) devices such as IEEE 802.11b devices implementing either the spread spectrum or frequency hopping mode.

Estimation is mainly based on measurements conducted on each frequency or channel in order to determine the presence of interference. Several methods are available ranging from BER, RSSI, packet loss rate, and negative ACKs. In this discussion, we show how estimation can be performed using BER measurements. However, the same remains true if other measurements were to be used instead. In a nutshell, BER based channel estimation works as follows. Each Bluetooth receiver maintains a *Channel Classification Table* where a BER measurement, BER_f, is associated to each frequency, f, as shown in Figure 1. Frequencies are classified "good" or "bad" depending on whether their corresponding BER is below or above a threshold value, BER^T, respectively.

Status	Frequency Offset	BER_f
	0	10^{-3}
	1	10^{-1}
	2	10^{-2}
	3	10^{-1}
	...	
	76	10^{-4}
	77	10^{-3}
	78	10^{-3}

Good
Bad

Fig. 1. Channel Classification Table

Depending on the type of measurements used, the receiver may have to explicitely send the measurements collected or the channel classification table to the transmitter. In the case where negative ACKs are used, the transmitter can construct a channel classfication table for the receiver without an explicit message exchange.

An interesting question we would like to answer is how to adapt the channel estimation procedure to changes in the environment. Mainly, we would like to determine (1) the time it takes to perform the estimation, (2) and how often it is performed.

First, we define two types of channel estimation procedures. **Offline** estimation refers to a period of time where packets are sent on all frequencies regardless of their classification. It takes place every interval, EI, where during a window, EW, all frequencies in the sequence are used to transmit data if available. On the other hand, **online** estimation is performed continuously while scheduling is performed. The two methods are complementary. Online estimation is mainly used to determine what frequencies the transmitter should avoid using, while offline estimation is used to bring "good" frequencies back in use.

For online estimation, we use a moving window average to compute BER. At time $t+1$, the BER associated with frequency f is updated as follows:

$$BER^f_{t+1} = \alpha \times BER^f_{t+1} + (1-\alpha) \times BER^f_t \quad (1)$$

where $0 < \alpha \leq 1$. When α is equal to 1, no history is kept and the update is instantaneous.

Next, we give a lower bound for the time it takes to perform the offline estimation and describe how to adjust the estimation interval, EI, based on the environment's dynamics.

A. Channel Estimation Time

The time to perform the offline estimation depends on the frequency hopping rate since the methods used to perform the classification depend on BER measurements per frequency visited. A lower bound calculation is as follows. First, we assume a hop rate of 1600 hops/s given single slot packets. For each receiver the hopping rate is 1600/2 hops/s, or 800 hops/s since nodes receive on every other frequency in the hopping pattern. Next, we consider the Bluetooth frequency hopping algorithm. In a window of 32 frequencies, every frequency is selected once, then the window is advanced by 16 frequencies, and the process is repeated. Therefore, it takes 5 windows of 32 frequencies in order to visit each of the 79 frequencies twice. In other words, 160 hops visit each frequency twice. The time to visit each frequency one time per receiver is 160/800/2 = 0.1 seconds or 100 ms. In fact, 100 ms constitutes a lower bound in case multi-slot packets are used or in case more than one measurement per frequency is desired.

B. Channel Estimation Interval

How often to update the channel estimation depends on the application and the dynamics of the scenario used. We propose an adaptive procedure to adjust EI, which the interval between two consecutive offline channel estimation measurements. First, we let δ, be the percentage of frequencies that change classification status (from "good" to "bad" or vice versa) during the previous offline estimation. Initially, EI is set to EI_{min}. Then, EI is updated every interval, k, according to the rationale that if a change were to happen it is likely to happen again in the near future and therefore EI is set to EI_{min}. Otherwise, the window is doubled.

$$\begin{aligned} EI_{k+1} &= max\,(2 * EI_k, EI_{max}); \qquad if\ \delta \leq 0.1 \\ EI_{k+1} &= EI_{min} \qquad otherwise \end{aligned} \quad (2)$$

C. Bluetooth Scheduling

We propose a scheduling policy that makes use of the information available in the channel classification table in order to avoid packet transmission in a "bad" receiving channel. This so-called Bluetooth Interference Aware Scheduling (BIAS) was first introduced in [4]. It consists of at least three components, namely a channel estimation procedure, a credit function that allocates bandwidth to each device according to its service requirements, and a priority scheduling function. Given that the master device controls all transmission in the piconet, this policy needs to only be implemented in the master device. Furthermore, since a slave transmission follows a master transmission, the master transmits in slot after it verifies that both the slave's receiving frequency and its own receiving frequency are "good". Otherwise, the master skips the current transmission slot and repeats that procedure over again in the next transmission opportunity.

We let u_i be the probability that a pair of master/slave transmission slots are "good". u_i represents the available spectrum to slave S_i. Therefore, we write:

$$\begin{aligned} u_i \quad = \quad & min((1-1/79), P(slave\ i\ has\ a\ good\ receiving\ frequency) \\ & \times P(master\ has\ a\ good\ receiving\ frequency)) \end{aligned} \quad (3)$$

where

$$P(device\ i\ has\ a\ good\ receiving\ frequency) = \frac{Number\_of\_good\_Channels_i}{Total\_Number\_of\_Channels} \quad (4)$$

We define c^i_{up} and c^i_{dn} for the upstream and downstream credits respectively according to the following:

$$\begin{aligned} c^i_{up} \quad &= \quad \gamma^i_{up} \times N \\ c^i_{dn} \quad &= \quad \gamma^i_{dn} \times N \end{aligned} \quad (5)$$

where N is the number of slots considered in the allocation, and $\gamma^i_{up/dn}$ is the bandwidth allocated to a Slave i in the upstream/downstream direction (as a percentage of the capacity). Devices with a positive credit counter, c_i, are allowed to send data packets. POLL and NULL packets can always be sent regardless of the value of the credit counters. The other component of the algorithm is to give a priority of access to certain devices. We use a two-tier system with high and low priorities. A high priority is used to support delay constrained application such as voice and MP3, while a low priority is used to support best effort connections such as ftp, and http. Thus, high priority connections are serviced first, and low priority connections second. Also, among connections of the same priority, receivers with fewer number of "good" channels are given priority over other receivers with a greater number of "good" channels. Thus, w^i_{up} and w^i_{dn} are defined as follows:

$$\begin{aligned} w^i_{up} \quad &= \quad c^i_{up} \times (1-u_i) \\ w^i_{dn} \quad &= \quad c^i_{dn} \times (1-u_i) \end{aligned} \quad (6)$$

The master schedules a data transmission for slave i such as to maximize the product of the weights in the up and downstream:

$$i = max^f_S (w^i_{up} \times w^i_{dn}) \quad (7)$$

To transmit a POLL packet, the master looks only at the weight function in the upstream:

$$i = max_S^f(w_{up}^i) \tag{8}$$

The selection of a slave is restricted over the set of slaves S that can receive on the master's current transmission frequency, f. Thus, any slave that experiences a high level of BER on the current transmission frequency is not considered. Four sets of slaves are formed, A_{data}^f, A_{poll}^f, B_{data}^f, and B_{poll}^f. A_{data} and A_{poll} represent the set of high priority connections requiring data and POLL packet transmissions respectively. Similarly, B_{data} and B_{poll} represent low priority connections. Thus, every master's transmission slot, the master invokes the scheduling algorithm. It first tries to schedule a packet to high priority slaves in group A, then a POLL packet, before it moves to group B. The credit counters and weights are updated accordingly after every master's transmission. Additional details on the algorithm's operation are given in [5].

III. Simulation Results

In this section, we present two sets of experiments to evaluate the channel estimation algorithm's responsiveness to changes in the environment. In the first experiment, traffic generation is based on a Poisson on-off traffic source for WLAN and Bluetooth. In the second experiment, we use more realistic traffic such as MPEG, voice, FTP and HTTP. Our simulation environment is based on a detailed MAC, PHY and channel models for Bluetooth and IEEE 802.11 (WLAN) as described in [6]. The parameters used in the setup vary according to the experiment. The common simulation parameters are summarized in Table I. The simulations are run for 1800 seconds of simulated time unless specified otherwise. We run 10 trials using a different random seed for each trial. In addition, to plotting the mean value, we verify that that the statistical variation around the mean values are very small (less than 1%).

TABLE I
COMMON SIMULATION PARAMETERS

Bluetooth Parameters	**Values**
ACL Baseband Packet Encapsulation	DH1, DH5
Transmitted Power	1 mW
WLAN Parameters	**Values**
Packet Interarrival Time	2.172 ms
Offered Load	60 % of Channel Capacity
Transmitted Power	25 mW
Data Rate	11 Mbits/s
PLCP Header	192 bits
Packet Header	224 bits
Payload Size	12000 bits

The performance metrics include the **packet loss**, the **mean access delay**, and the channel **estimation reponse time**. The packet loss is the percentage of packets dropped due to interference over the total number of packets received. The access delay measures the time it takes to transmit a packet from the time it is passed to the MAC layer until it is successfully received at the destination. The delay is measured at the L2CAP layer. The channel estimation response time measures the time to track changes in the channel state. The time to avoid using a frequency that becomes "bad" is denoted by RT_{GB}^f, while the time to start reusing a frequency after it has become "good" is RT_{BG}^f. In addition, to the response time measured per frequency, we measure the time it takes to scan the entire spectrum. This measure is denoted by RT_{GB}^S and RT_{BG}^S for finding all "bad" frequencies and "good" frequencies respectively.

We use the topology illustrated in Figure 2 that consists of a WLAN system (a source-sink pair), and one Bluetooth piconet with one master and one slave device. The WLAN sink-source devices are 15 meters apart while the Bluetooth devices are 2 meters apart. The distance between the WLAN transmitter (sink) and the Bluetooth devices is around 1.4 meters.

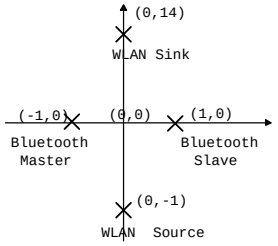

Fig. 2. Topology for Experiments 1 and 2

A. Experiment 1

For Bluetooth, a generic source that generates either $DH1$ or $DH5$ packets is considered. The packet interarrival mean time in seconds, t_B, is exponentially distributed and is computed according to

$$t_B = 2 \times l \times 0.000625 \times (\frac{1}{\lambda} - 1) \tag{9}$$

where l is the packet length in slots and λ is the offered load. We assume that WLAN is operating in the Direct Sequence Spread Spectrum (DSSS) mode. The WLAN source is transmitting data packets to the sink which is responding with ACKs. The WLAN packet payload is set to 12000 bits transmitted at 11 Mbits/s, while the PLCP header of 192 bits is transmitted at 1 Mbits/s. The packet interarrival time in seconds, t_W, is exponentially distributed and its mean is computed according to

$$t_W = (\frac{192}{1000000} + \frac{12224}{11000000})/\lambda \tag{10}$$

The offered load for Bluetooth is varied between 5 and 100%, while for WLAN the offered load is set to 60%. For Bluetooth, both DH1 (1 slot) and DH5 (5 slots) packets are used in order to compare the difference in transient times. In addition, the time the WLAN connection is on, T_{ON}, is exponentially distributed with a mean equal to 30 seconds, while the time the WLAN connection is off, T_{OFF}, is also exponentially distributed with mean equal to 60 seconds. In addition, we set $EI_{min} = 5$ seconds, $EI_{max} = 900$ seconds, and $\alpha = 0.9$. The channel estimation is performed during a window EW equal to 400 ms.

Results - Figure 3(a) and (b) give the packet loss and access delay respectively measured at the Bluetooth slave device. The packet loss when BIAS is used is negligible (less than 0.5%) for both DH1 and DH5 packets, while it is almost two orders of magnitude larger when no scheduling is used. The delay for DH1 packets is slighly higher with BIAS (about 1 ms) due to the extra time in delaying a transmission. This penalty disappears

for DH5 packets where avoiding the transmission in a "bad" frequency eliminates the delay associated with the retransmission of a corrupted packet. Figure 4(a) and (b) give the channel estimation response time per frequency and over the entire spectrum using DH1 and DH5 packets. Figure 4(a) gives $RT_{BG}^{f,S}$. The response time increases with respect to the offered load for both packet types. It starts at about 50 ms and goes up to 25 seconds. It is very much dependent on the estimation interval time EI and in theory could be much larger than RT_{GB}. In Figure 4(b) illustrating $RT_{GB}^{f,S}$, observe that the response time to find all bad frequencies in the spectrum converges to around 125 ms for both DH5 and DH1 packets at 100% offered load. This is close to the expected value of 100 ms derived in section II. Also, note that, as the offered load increases, R_{GB}^{S} is decreased since more packets are sent over the medium and frequencies can be scanned faster. The use of DH5 packets leads to a slower hopping rate and therefore increases the response times, up to 800 ms for an offered load of 10%. R_{GB}^{S} drops to about 150 ms at 100%. R_{GB}^{f} is around 20 ms for both packet types.

B. Experiment 2

In this experiment, we consider four application profiles, namely, voice, MP3, Ftp, and Http. The parameters describing these profiles are given in Table II and are based on the analysis of packet traces for FTP [7][8] and Http [9] [10].

TABLE II
APPLICATION PROFILE PARAMETERS

Parameters	Distribution	Value
Voice		
Packet Size (bytes)	Constant	60
Packet Interarrival (seconds)	Constant	0.02
MP3		
Packet Size (bytes)	Constant	462
Packet Interarrival (seconds)	Constant	0.026
Ftp		
Burst size (bytes)	Pareto	minimum = 10000, $\alpha = 0.9$
Connection Interarrival (seconds)	Exponential	60
Http		
(Source) Request Size (bytes)	Constant	240
(Sink) Reply Size (bytes)	Pareto	minimum=1000, $\alpha = 1.1$
Reply/Request Interarrival (seconds)	Weibull	k=0.5, $\theta = 1.5$
Connection Interarrival (seconds)	Pareto	minimum=60, $\alpha = 0.9$
Connection Duration (seconds)	Weibull	k=0.9, $\theta = e^{4.4}$

Results - The results in Tables III, IV, and V are consistent with the results discussed in the previous experiment. First, note that the packet loss in Table III is in the order of 10^{-3} or less as shown in Figure 3. Also, RT_{BG} is generally higher than RT_{GB} as observed in Experiment 1. Bringing back frequencies that become "good" can take more time. RT_{GB}^{f} starts at 5 ms for voice traffic and goes up to 130 ms for http and voice traffic for Bluetooth and WLAN respectively. The largest time to scan the entire spectrum is for HTTP and FTP traffic.

RT_{BG}^{f} varies between 50 ms to about 1 seconds. RT_{BG}^{S} varies between 200 ms to 26 seconds. This is consistent with the behavior exhibited in the previous experiment.

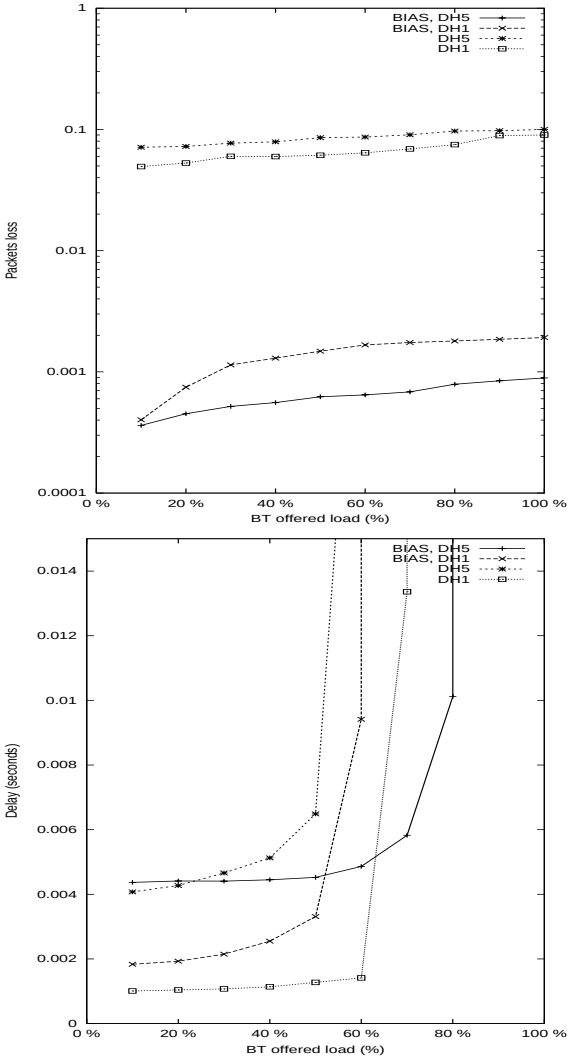

Fig. 3. $\frac{(a)}{(b)}$ Experiment 1. Variable Bluetooth Offered Load. (a) Probability of Packet Loss. (b) Mean Access Delay

IV. CONCLUDING REMARKS

This paper discusses a channel estimation technique used for Bluetooth to detect the presence of an interfering WLAN system. We show that combining a dynamic channel estimation method with scheduling a packet transmission in a "good" frequency helps mitigate the effect of interference. In addition, our results indicate that packet loss is less than 0.5% , while throughput is increased (lower delays), especially for longer Bluetooth packets. The algorithm's responsiveness is measured in terms of the time it takes to detect a "bad" channel which is around 20 ms. On the other hand, the time it takes to start reusing a frequency can take as long as 25 seconds. We plan to investigate other variations of channel estimation and Bluetooth scheduling mechanisms for other interference scenarios and analyze the impact of interference mitigation techniques on the performance of higher layer protocols.

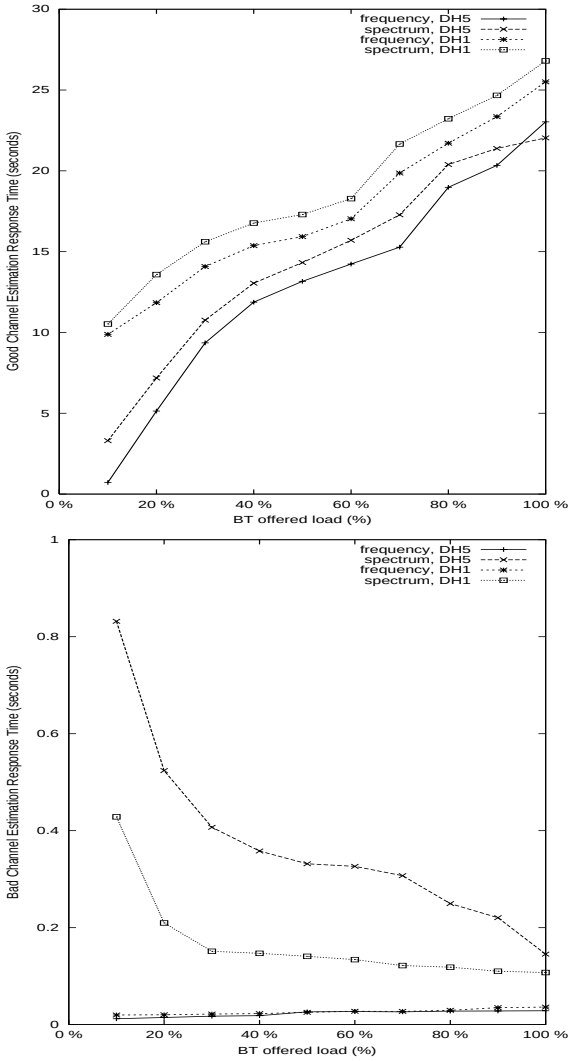

Fig. 4. $\frac{(a)}{(b)}$ Experiment 1. Variable Bluetooth Offered Load. (a) Time to Estimate a Good Channel, R_{GB} (b) Time to Estimate a Bad Channel, R_{BG}

ACKNOWLEDGEMENTS

The author would like to thank Olivier Rebala for his help in obtaining the simulation results.

REFERENCES

[1] J. Lansford, R. Nevo, E. Zehavi, "MEHTA: A method for coexistence between co-located 802.11b and Bluetooth systems," in *IEEE P802.11 Working Group Contribution, IEEE P802.15-00/360r0*, November 2000.

[2] B. Treister, A. Batra, K.C. Chen, O. Eliezer, "Adapative Frequency Hopping: A Non-Collaborative Coexistence Mechanism," in *IEEE P802.11 Working Group Contribution, IEEE P802.15-01/252r0*, Orlando, FL, May 2001.

[3] N. Golmie, and N. Chevrollier, "Techniques to Improve Bluetooth Performance in Interference Environment," in *Proceedings of MILCOM'01*, McLean, Virginia, October 2001.

[4] N. Golmie, N. Chevrollier, and I. Elbakkouri, "Interference Aware Bluetooth Packet Scheduling," in *Proceedings of GLOBECOM'01*, San Antonio, TX, November 2001.

[5] N. Golmie, " Bluetooth Dynamic Scheduling and Interference Mitigation," in *submitted for review to ACM/MONET'02*, 2002.

[6] N. Golmie, R.E. Van Dyck and A. Soltanian, "Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation," in *Proceedings of the Fourth ACM International Workshop on Modeling, Analysis, and Simulation of Wireless and Mobile Systems, MSWIM'01*, Rome, Italy, July 2001.

[7] M. Crovella, A. Bestavros , "Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes," in *IEEE/ACM Transactions on Networking*, December 1997, vol. 5, pp. 835–846.

[8] V. Paxon, and S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," in *IEEE/ACM Transactions on Networking*, June 1995, vol. 3, pp. 226–244.

[9] S. Deng , "Empirical model of WWW document arrivals at access link ," in *Proceedings of IEEE International Conference on Communications ICC'96*, 1996, vol. 3, pp. 1797–1802.

[10] B. Mah, "An Empirical Model of HTTP Network Traffic," in *Proceedings of INFOCOM'97*, 1997.

TABLE III

BLUETOOTH PACKET LOSS ($\times 10^{-4}$)

BT Traffic	WLAN Traffic			
	FTP	HTTP	MP3	Voice
FTP	8.3	8.7	5.4	3
HTTP	10	7.5	5.4	11
MP3	0.43	1	0.5	27
Voice	7.5	5.5	11	16

TABLE IV

(BAD) CHANNEL ESTIMATION RESPONSE TIME (SECONDS), (RT^f_{GB},RT^S_{GB})

BT Traffic	WLAN Traffic			
	FTP	HTTP	MP3	Voice
FTP	(0.02,0.33)	(0.026,0.38)	(0.057,0.62)	(0.09,0.78)
HTTP	(0.04,0.11)	(0.06,0.15)	(0.07,0.21)	(0.13,0.31)
MP3	(0.03,1.12)	(0.023,2.05)	(0.06,0.15)	(0.0055,0.18)
Voice	(0.04,0.13)	(0.10,0.20)	(0.056,0.35)	(0.08,0.64)

TABLE V

(GOOD) CHANNEL ESTIMATION RESPONSE TIME (SECONDS), (RT^f_{BG},RT^S_{BG})

BT Traffic	WLAN Traffic			
	FTP	HTTP	MP3	Voice
FTP	(0.17,23.39)	(0.22,14.64)	(0.29,1.43)	(0.74,10.68)
HTTP	(0.07,26.39)	(0.11,21.61)	(0.10,1.28)	(0.11,0.65)
MP3	(0.75,8.80)	(1.17,23.61)	(0.1,0.2)	(1.09,5.87)
Voice	(0.05,29.22)	(0.09,15.31)	(0.097,2.27)	(0.10,0.83)